

REMARKS

In the Office Action, the Examiner rejected Claims 1, 4-7, 11, 12, 15, 17-19 and 21, which were all of the then pending claims, under 35 U.S.C. 103 as being unpatentable over the prior art. In particular, Claims 1, 4-7, 15 and 17-19 were rejected as being unpatentable over U.S. Patent 6,957,421 (Hundt, et al.) in view of a paper titled "Checking System Rules Using System-Specific, Programmer-Written Compiler Extensions" (Engler, et al.). Claims 11, 12 and 21 were rejected as being unpatentable over U.S. Patent 6,721,941 (Morshed) in view of U.S. Patent 6,161,196 (Tsai).

It is noted that the previous rejection of Claims 1, 4-7, 15 and 17-19 under 35 U.S.C. 102 as being fully anticipated by U.S. Patent 6,721,941 (Morshed); and the previous rejection of Claims 21, 11 and 12 were rejected under 35 U.S.C. 103 as being unpatentable over Morshed in view of U.S. Patent 6,839,893 (Bates), were withdrawn.

Independent Claims 1, 15 and 21 are being amended to better define the subject matters of these claims. New Claims 23 and 24, which are dependent from Claim 22, are being added to describe features of an embodiment of the invention.

For the reasons set forth below, Claims 1, 4-7, 11, 12, 15, 17-19, 21, 23 and 24 patentably distinguish over the prior art and are allowable. The Examiner is thus respectfully requested to reconsider and to withdraw the rejection of Claims 1, 4-7, 11, 12, 15, 17-19 and 21 under 35 U.S.C. 103, and to allow these claims and new Claims 23 and 24.

Generally, Claims 1, 4-7, 11, 12, 15, 17-19, 21, 23 and 24 patentably distinguish over the prior art because the prior art does not disclose or render obvious the way in which the pattern detectors are used, as described in independent Claims 1, 15 and 21, to detect violations of

defined coding patterns. In particular, the prior art does not disclose or render obvious the feature that each of these pattern detectors, when invoked by the pattern detector manager, which occurs upon hitting one of the main entry breakpoints in the computer program, inserts a plurality of additional breakpoints into the computer program at specific points in the computer program associated with one of the coding patterns to detect violations of that coding pattern.

In order to best understand this feature and its significance, it may be helpful to review briefly this invention and the prior art.

The present invention, generally, relates to the automatic detection of problematic coding patterns and violations of best practices patterns at program runtime. As discussed in detail in the present application, in any large software deployment, subtle coding defects can cause problems in successful deployment of the software. Tracking down these defects may be extremely difficult in the production environment because of a number of factors.

One factor is that, in many cases, the symptoms are usually not unique to a particular type of software defect. This makes correlating symptoms to specific defects nearly impossible. Another factor is that many symptoms may manifest themselves only during production level loads and production configurations. This means that the defects are often undetected in the testing and debugging of software. In addition, the reasons why a piece of code is defective are often complex and may span third party libraries and frameworks.

The present invention effectively addresses these issues. Generally, this is done by observing the behavior of a running program within the context of a large number of defined coding patterns, and automatically flagging violations of the coding patterns when they occur. More specifically, in accordance with one embodiment of the invention, a software tool is

provided for monitoring the behavior of a running computer program for code patterns that violate a given set of coding rules. This software tool comprises a pattern detector manager and a plurality of individual pattern detectors. Each of these pattern detectors is used to detect a violation of one of the coding patterns. Each of these coding patterns includes a plurality of steps, and each of these steps is associated with a defined location in the computer program.

The pattern detector manager is provided for inserting a main entry breakpoint at one or more defined points in the computer program. When any of these breakpoints are hit during the running of the computer program, the pattern detector manager invokes each of the pattern detectors to insert the plurality of additional breakpoints into the computer program at the defined locations in that program that are associated with the steps of the coding pattern the detecting pattern is used to detect. These pattern detectors track the inserted, additional breakpoints to detect violation of those defined coding patterns.

The prior art does not disclose or render obvious the above-described way in which the additional breakpoints are inserted into the computer program and to detect violations of the coding patterns.

For example, Hundt, et al. discloses a procedure for providing a debugging capability for program code instrumentation. In one disclosed embodiment, an instrumentor inserts an instrumentation breakpoint at the beginning of a block of original code; and when this breakpoint is reached during execution of the application program, the instrumentor generates a block of instrumented code. This block of instrumented code may include debugging breakpoints that are carried from the original code or are inserted into the block of instrumented code during debugging. The instrumented code can then be executed until debugging breakpoints are

reached that stop the program flow, allowing a programmer to perform debugging functions at these debugging breakpoints.

Morshed, et al. discloses a method and system for monitoring execution of an application in a computer system. In the procedure described in this reference, debugging information is reported to a program monitor using an application debugging interface, and program execution information is gathered using the debugging information to monitor execution of the application.

One specific operation of a virtual machine is discussed in columns 21-23 of Morshed, et al. In this operation, a computer code is tested to see if various lines or instructions have been reached. If these tests are positive, various instruments are instrumented, as represented in Fig. 14.

There are a number of important differences between the above-discussed embodiment of the present invention and the procedures disclosed in Hundt, et al. and Morshed, et al.

One important difference is the way in which the pattern detectors are used. Specifically, with the present invention, as described above, each of the pattern detectors is used to detect a violation of an associated coding pattern. Also, each of the coding patterns includes a plurality of steps, and each of these steps is associated with a defined location in the computer program. When one of the pattern detectors is invoked, that detector inserts a plurality of additional breakpoints into the computer program. Each of these additional breakpoints is inserted into the program at the locations associated with the steps of the coding pattern that the detector is used to detect.

Neither Hundt, et al. nor Morshed, et al. operates in this way.

The other references of record have been reviewed, and these other references, whether considered individually or in combination, also do not disclose or suggest the use of the pattern detectors, as described above.

The Engler paper describes a plurality of pattern detectors that are used to detect a plurality of coding patterns, but this paper does not disclose or render obvious inserting breakpoints as is done in the present invention.

Tsai discloses an indirect software instrumentation technique to achieve fault tolerance in a computing system. In this technique, different copies of a given target program are executed on different machines in the system. If a fault is detected in one of the copies, a checkpoint is taken of another copy that has been determined to be fault-free, and a new copy is restarted from the checkpoint.

Tsai was cited for its disclosure of detecting the non-occurrence of an event. This reference does not disclose, though, detecting patterns by inserting breakpoints into the computer program in the way that is done with the present invention.

Independent Claims 1 and 15 are being amended to describe the above-discussed feature of an embodiment of the invention. More specifically, these claims are being amended to describe the features that each of the coding patterns includes a plurality of steps, and that each of these steps is associated with a defined location in the computer program. Claims 1 and 15, as presented herewith, also include the limitation that upon hitting one of the main entry breakpoints in the computer program, the pattern detector manager invokes each of the pattern detectors to insert the plurality of additional breakpoints into the computer program at the plurality of defined locations in the computer program associated with the steps of the coding

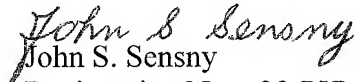
pattern associated with said each pattern detector. Claims 1 and 15 further describe the feature that the pattern detectors track these inserted, additional breakpoints to detect violations of those defined coding patterns.

Independent Claim 21 is directed to a method of detecting code patterns in a computer program that violate a given set of coding rules, and this claim now includes limitations similar to those in Claims 1 and 15. For instance, Claim 21 includes the limitation that each coding rule of the set of coding rules includes a plurality of steps, and each of these steps is associated with a defined location in the computer program. Claim 21 also sets forth the limitation that upon hitting one of the main entry breakpoints in the computer program, the pattern detector manager invokes each of the pattern detectors to insert the plurality of additional breakpoints into the computer program at the plurality of defined locations in the computer program associated with the steps of the coding rule associated with said each pattern detector.

In light of the above-discussed differences between Claims 1, 15 and 21 and the prior art, and because of the advantages associated with those differences, Claims 1, 15 and 21 patentably distinguish over the prior art and are allowable. Claims 4-7, 23 and 24 are dependent from Claim 1 and are allowable therewith. Also, Claims 17-19 are dependent from, and are allowable with, Claim 15, and Claims 11 and 12 are dependent from Claim 21 and are allowable therewith. Accordingly, the Examiner is asked to reconsider and to withdraw the rejection of Claims 1, 4-7, 11, 12, 15, 17-19 and 21 under 35 U.S.C. 103, and to allow these claims and new Claims 23 and 24.

For the reasons set forth above, the present application is in condition for allowance, a notice of which is requested. If the Examiner believes that a telephone conference with Applicants' Attorneys would be advantageous to the disposition of this case, the Examiner is asked to telephone the undersigned.

Respectfully submitted,


John S. Sensny
Registration No.: 28,757
Attorney for Applicants

SCULLY, SCOTT, MURPHY & PRESSER, P.C.
400 Garden City Plaza – Suite 300
Garden City, New York 11530
(516) 742-4343

JSS:jy